

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA
MC68030

Technical Summary

Second Generation

32-Bit Enhanced Microprocessor

The MC68030 is the industry's second generation 32-bit enhanced microprocessor. The MC68030 is a virtual memory microprocessor based on an MC68020 core with additional enhanced performance features. Increased internal parallelism is provided by multiple internal data buses and address buses and a versatile bus controller that supports two-clock cycle bus accesses and one-clock cycle burst accesses in order to maximize performance with paged mode, nibble mode, and static column DRAM technology. A 256-byte on-chip instruction cache in addition to a 256-byte data cache improves data flow to the execution unit and further boosts performance. On-chip paged memory management reduces the minimum physical bus cycle time to two clocks, and provides zero translation time to any bus cycle. The paged memory management structure can be enabled/disabled by software for applications not requiring the memory management feature. The rich instruction set and addressing modes of the MC68020 have been maintained allowing a clear migration path for M68000 systems.

The main features of the MC68030 are:

- Object Code Compatible with the MC68020 and Earlier M68000 Microprocessors
- Complete 32-Bit Non-Multiplexed Address and Data Buses
- Sixteen 32-Bit General Purpose Data and Address Registers
- Two 32-Bit Supervisor Stack Pointers and 10 Special Purpose Control Registers
- 256-Byte Instruction Cache and 256-Byte Data Cache that can be Accessed Simultaneously
- Paged Memory Management Unit that Translates Addresses in Parallel with Instruction Execution
- Two Transparent Segments Allow Untranslated Blocks to be Defined for Systems that Transfer Large Blocks of Data to Predefined Addresses, e.g., Graphics Applications
- Pipelined Architecture with Increased Parallelism Allows Accesses from Internal Caches to Occur in Parallel with Bus Transfers and Multiple Instructions to be Executing Concurrently
- Enhanced Bus Controller Supports Asynchronous Bus Cycles, Synchronous Bus Cycles that can Operate in Two Clocks, and Burst Data Transfers that can Operate in One Clock, all with Physical Addresses
- Dynamic Bus Sizing Supports 8-/16-/32-Bit Memories and Peripherals
- Complete Support for Coprocessors with the M68000 Coprocessor Interface
- 4-Gigabyte Direct Addressing Range
- Implemented in Motorola's HCMOS Technology that Allows CMOS and HMOS (High Density NMOS) Gates to be Combined for Maximum Speed, Low Power, and Small Die Size
- Selection of Processors Speeds: 16.67 and 20 MHz

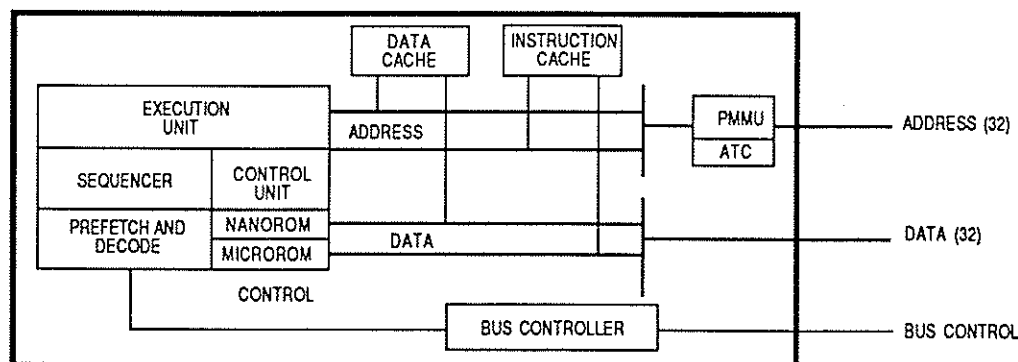


Figure 1. MC68030 Block Diagram

This document contains information on a new product. Specifications and information herein are subject to change without notice.



INTRODUCTION

The MC68030 is an enhanced 32-bit HCMOS microprocessor that incorporates the capabilities of the MC68020 MPU, an on-chip data cache, an on-chip instruction cache, an improved bus controller, multiple internal data buses, multiple internal instruction buses, and an on-chip paged memory management structure defined by the MC68851 Paged Memory Management Unit. The MC68030 maintains the 32-bit registers available with the entire M68000 Family as well as the 32-bit address and data paths, rich instruction set, versatile addressing modes, and flexible coprocessor interface provided with the MC68020. In addition, the internal operations of this integrated processor are designed to operate in parallel, allowing multiple instructions to be executed concurrently. It allows instruction execution to proceed in parallel with accesses to the internal caches, the on-chip memory management unit, and the bus controller.

The MC68030 fully supports the non-multiplexed asynchronous bus of the MC68020 as well as a dynamic bus sizing mechanism that allows the processor to transfer operands to or from external devices while automatically determining device port size on a cycle-by-cycle basis. In addition to the asynchronous bus, the MC68030 also supports a fast synchronous bus for off-chip caches and fast memories. Further, the MC68030 bus is capable of fetching up to four long words of data in a burst mode compatible with DRAM chips that have burst capability. Burst mode can reduce by up to 50% the time necessary to fetch the four long words. The four long words are used to prefill the on-chip instruction and data caches so that the hit ratio of the caches improves and the average access time for operand fetches is minimized.

The block diagram shown in Figure 1 depicts the major sections of the MC68030 and illustrates the autonomous nature of these blocks. The bus controller consists of the address and data pads, the multiplexors required to support dynamic bus sizing, and a macro bus controller which schedules the bus cycles on the basis of priority. The micromachine contains the execution unit and all related control logic. Microcode control is provided by a modified two-level store of microrom and nanorom contained in the micromachine. Programmed logic arrays (PLAs) are used to provide instruction decode and sequencing information. The instruction pipe and other individual control sections provide the secondary decode of instructions and generate the actual control signals that result in the decoding and interpretation of nanorom and microrom information.

The instruction and data cache blocks operate independently from the rest of the machine, storing information read by the bus controller for future use with very fast access time. Each cache resides on its own address and data buses, allowing simultaneous access to both. Both the caches are organized as 64 long word entries (256 bytes) with a block size of four long words. The data cache uses a write-through policy with no write allocation on cache misses.

Finally, the memory management unit controls the mapping of addresses for page sizes ranging from 256 bytes to 32K bytes. Mapping information stored in descriptors resides in translation tables in memory that are

automatically searched by the MC68030 on demand. Recently-used descriptors are maintained in a 22-entry fully associative cache called the Address Translation Cache (ATC) allowing address translation and other MC68030 functions to occur simultaneously. Additionally, the MC68030 contains two transparent translation registers that can be used to define a one-to-one mapping for two segments ranging in size from 16M bytes to 4G bytes each.

PROGRAMMING MODEL

As shown in the programming models (Figures 2 and 3), the MC68030 has sixteen 32-bit general purpose registers, a 32-bit program counter, two 32-bit supervisor stack pointers, a 16-bit status register, a 32-bit vector base register, two 3-bit alternate function code registers, two 32-bit cache handling (address and control) registers, two 64-bit root pointer registers used by the MMU, a 32-bit translation control register, two 32-bit transparent translation registers, and a 16-bit MMU status register. Registers D0-D7 are used as data registers for bit and bit field (1 to 32 bit), byte (8 bit), word (16 bit), long word (32 bit), and quad word (64 bit) operations. Registers A0-A6 and the user, interrupt, and master stack pointers are address registers that may be used as software stack pointers or base address registers. In addition, the address registers may be used for word and long word operations. All of the 16 (D0-D7, A0-A7) registers may be used as index registers.

The status register (Figure 4) contains the interrupt priority mask (three bits) as well as the condition codes: extend (X), negate (N), zero (Z), overflow (V), and carry (C). Additional control bits indicate that the processor is in the trace mode (T1 or T0), supervisor/user state (S), and master/interrupt state (M).

All microprocessors of the M68000 Family support instruction tracing (via the T0 status bit in the MC68030) where each instruction executed is followed by a trap to a user-defined trace routine. The MC68030 also has the capability to trace only on the change of flow instructions (branch, jump, subroutine call and return, etc.) using the T1 status bit. These features are important for software program development and debug.

The vector base register is used to determine the run-time location of the exception vector table in memory, hence it supports multiple vector tables so each process or task can properly manage exceptions independent of each other.

The M68000 Family processors distinguish address spaces as supervisor/user, program/data, and CPU space. These five combinations are specified by the function code pins, FC0-FC2, during bus cycles, indicating the particular address space. Using the function codes, the memory subsystem (hardware) can distinguish between supervisor mode accesses and user accesses as well as program accesses, data accesses, and CPU space accesses. Additionally, the system software can configure the on-chip MMU so that supervisor/user privilege checking is performed by the address translation mechanism and the look-up of translation descriptors can be differentiated on the basis of function code. To support the full

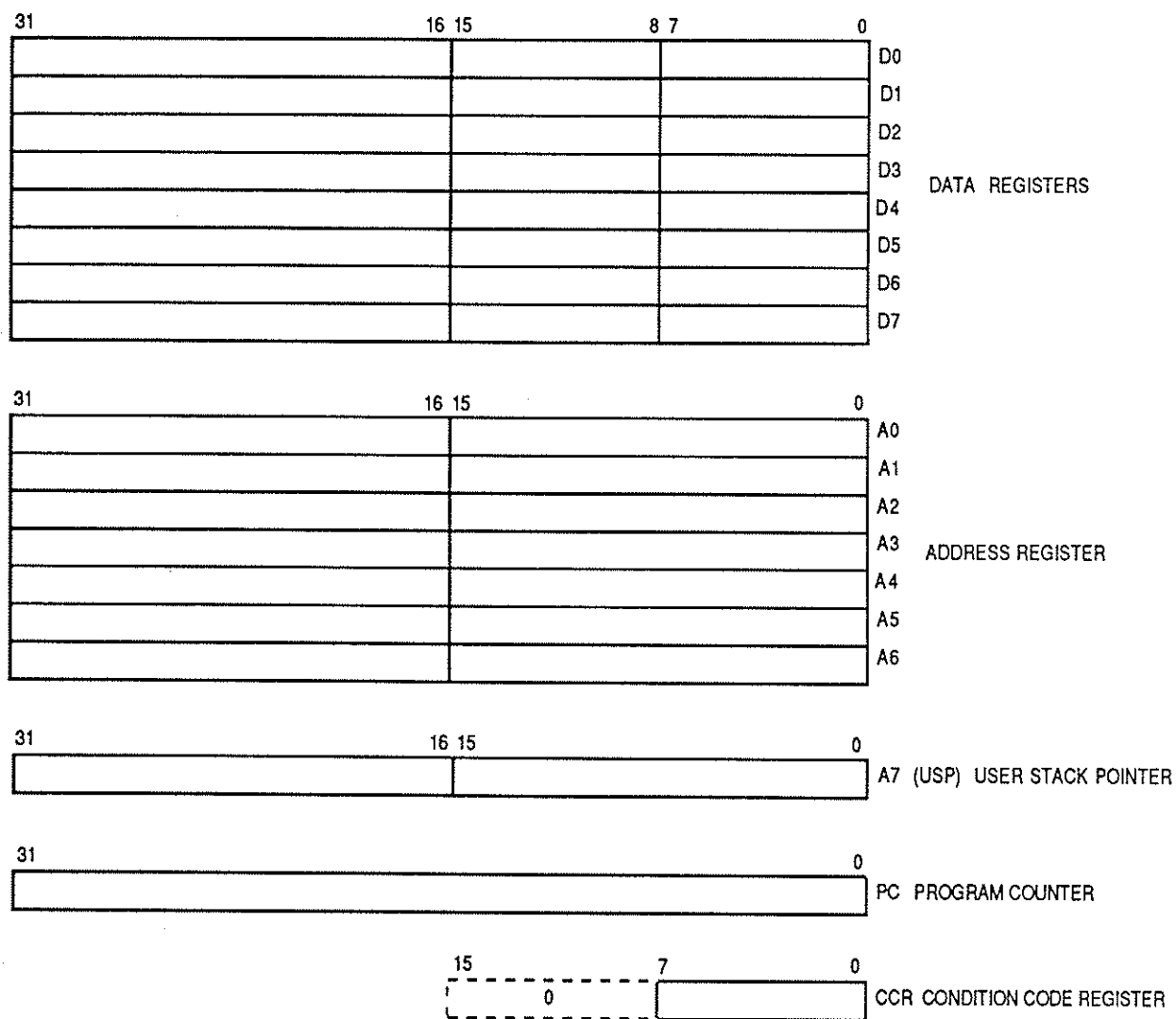


Figure 2. User Programming Model

privileges of the supervisor, the alternate function code registers allow the supervisor to specify the function code for an access by preloading the SFC/DFC registers appropriately.

The cache registers (control – CACR, address – CAAR) allow supervisor software manipulation of the on-chip instruction and data caches. Control and status accesses to the caches are provided by the cache control register (CACR), while the cache address register (CAAR) specifies the address for those cache control functions that require an address.

All of the MMU registers (CRP, SRP, TC, TT0, TT1, and PSR) are accessible by the supervisor only. The CPU root pointer contains a descriptor for the first pointer to be used in the translation table search for page descriptors pertaining to the current task. If the SRE (Supervisor Root

pointer Enable) bit of the translation control register is set, the supervisor root pointer is used as a pointer to the translation tables for all supervisor accesses. If the SRE bit is clear, this register is unused and the CPU root pointer is used for both supervisor and user translations. The translation control register configures the table look-up mechanism to be used for all table searches as well as the page size and any initial shift of logical address required by the operating system. In addition, this register has an enable bit that enables the MMU. The transparent translation registers can be used to define two transparent windows for transferring large blocks of data with untranslated addresses. Finally, the MMU status register (PSR) contains status information related to a specific address translation and the results generated by the PTEST instruction. This information can be useful in locating the cause of an MMU fault.

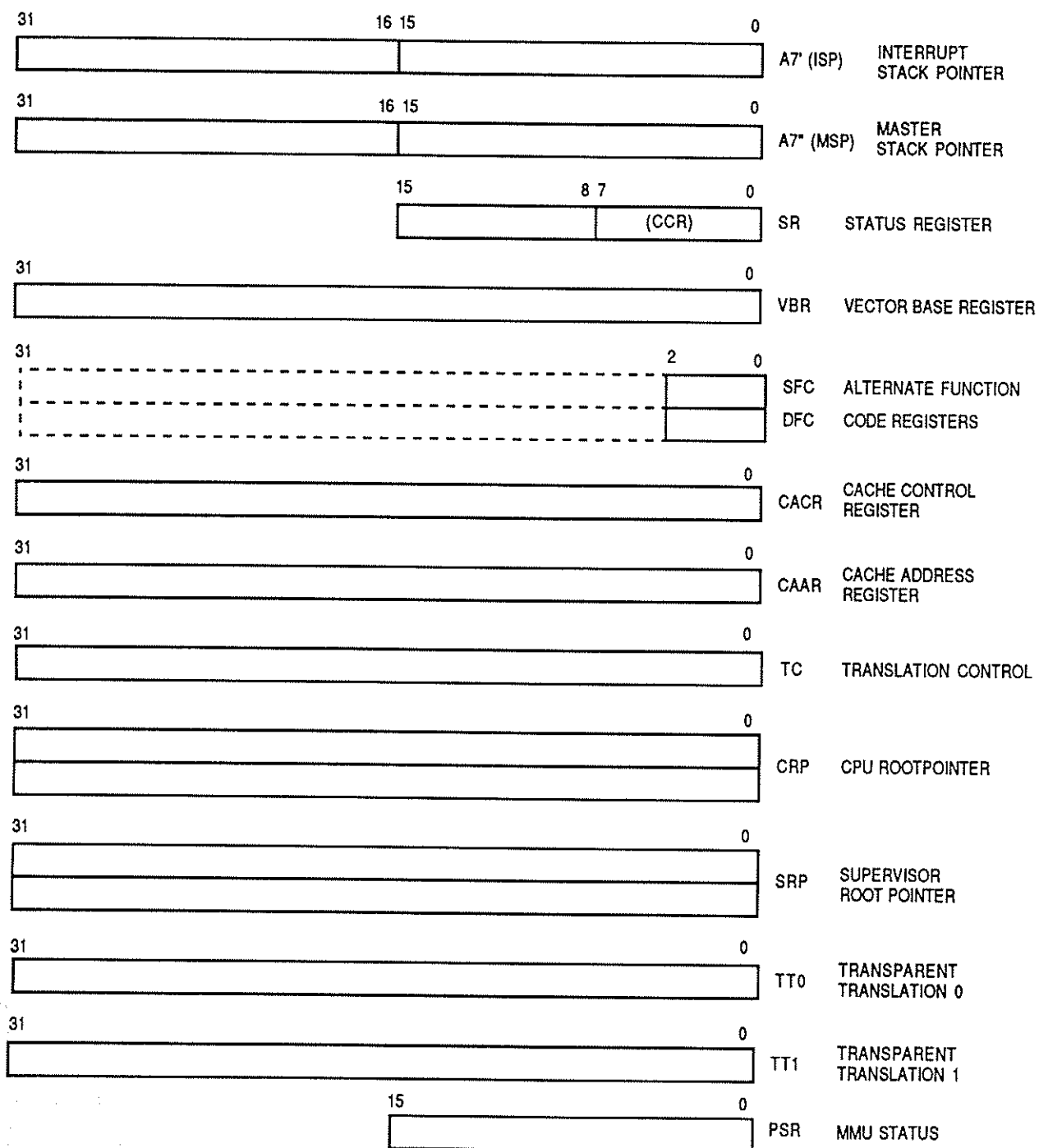


Figure 3. Supervisor Programming Model Supplement

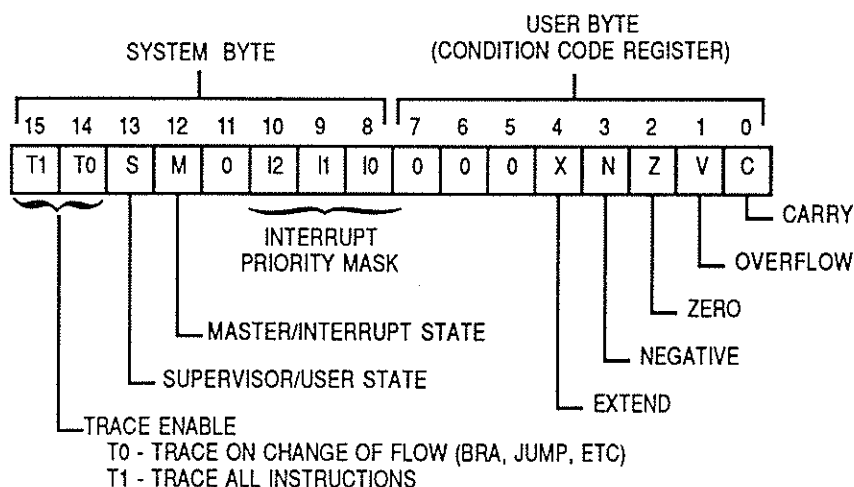


Figure 4. Status Register

DATA TYPES AND ADDRESSING MODES

Seven basic data types are supported on the MC68030. These are:

- Bits
- Bit Fields (String of consecutive bits, 1-32 bits long)
- BCD Digits (Packed: 2 digits/byte, Unpacked: 1 digit/byte)
- Byte Integers (8 bits)
- Word Integers (16 bits)
- Long Word Integers (32 bits)
- Quad Word Integers (64 bits)

In addition, operations on other data types, such as memory addresses, status word data, etc., are provided in the instruction set. The coprocessor mechanism allows direct support of floating-point data types with the MC68881 floating-point coprocessor, as well as specialized user-defined data types and functions.

The 18 addressing modes, shown in Table 1, include nine basic types:

- Register Direct
- Register Indirect
- Register Indirect with Index
- Memory Indirect
- Program Counter Indirect with Displacement
- Program Counter Indirect with Index
- Program Counter Memory Indirect
- Absolute
- Immediate

The register indirect addressing modes support postincrement, predecrement, offset, and indexing. These capabilities are particularly useful for handling advanced data structures common to sophisticated applications and high level languages. The program counter relative mode also has index and offset capabilities; programmers find that this addressing mode is required to support position-independent software. In addition to these addressing modes, the MC68030 provides index sizing and scaling; these features provide performance enhancements to the programmer.

INSTRUCTION SET OVERVIEW

The MC68030 instruction set is shown in Table 2. Each instruction, with few exceptions, operates on bytes, words, and long words, and most instructions can use any of the 18 addressing modes. The MC68030 is upward source- and object-level code compatible with the M68000 Family because it supports all of the instructions that previous family members offer. Included in this set are the bit field operations, binary coded decimal support, bounds checking, additional trap conditions, and additional multi-processing support (CAS and CAS2 instructions) offered by the MC68020. The new instructions supported by the MC68030 are a subset of the instructions introduced by the MC68851 paged memory management unit and are used to communicate with the MMU.

INSTRUCTION AND DATA CACHES

Studies have shown that typical programs spend most of their execution time in a few main routines or tight loops. This phenomenon is known as locality of reference, and has an impact on the performance of the program. The MC68010 takes limited advantage of this phenomenon with the loop mode of operation that can be used with the DBcc instruction. The MC68020 takes much more advantage of locality with its 256-byte on-chip instruction cache. The MC68030 takes further advantage of cache technology to provide the system with two on-chip caches, one for instructions and one for data.

MC68030 CACHE GOALS

Similar to the MC68020, there were two primary goals for the MC68030 microprocessor caches. The first design goal was to reduce the processor external bus activity even more than what was accomplished with the MC68020. The second design goal was to increase effective CPU throughput as larger memory sizes or slower

Table 1. MC68030 Addressing Modes

Addressing Modes	Syntax
Register Direct Data Register Direct Address Register Direct	Dn An
Register Indirect Address Register Indirect Address Register Indirect with Post Increment Address Register Indirect with Predecrement Address Register Indirect with Displacement	(An) (An) + - (An) (d ₁₆ ,An)
Register Indirect with Index Address Register Indirect with Index (8-Bit Displacement) Address Register Indirect with Index (Base Displacement)	(d ₈ ,An,Xn) (bd,An,Xn)
Memory Indirect Memory Indirect Post-Indexed Memory Indirect Pre-Indexed	((bd,An),Xn,od) ((bd,An,Xn),od)
Program Counter Indirect with Displacement	(d ₁₆ ,PC)
Program Counter Indirect with Index PC Indirect with Index (8-Bit Displacement) PC Indirect with Index (Base Displacement)	(d ₈ ,PC,Xn) (bd,PC,Xn)
Program Counter Memory Indirect PC Memory Indirect Post-Indexed PC Memory Indirect Pre-Indexed	((bd,PC),Xn,od) ((bd,PC,Xn),od)
Absolute Absolute Short Absolute Long	xxx.W xxx.L
Immediate	#(data)

NOTES:

Dn = Data Register, D0-D7

An = Address Register, A0-A7

d₈, d₁₆ = A two's-complement, or sign-extended displacement; added as part of the effective address calculation; size is 8 (d₈) or 16 (d₁₆) bits; when omitted, assemblers use a value of zero.

Xn = Address or data register used as an index register; form is Xn.SIZE*SCALE, where SIZE is .W or .L (indicates index register size) and SCALE is 1, 2, 4, or 8 (index register is multiplied by SCALE); use of SIZE and/or SCALE is optional.

bd = A two's-complement base displacement; when present, size can be 16 or 32 bits.

od = Outer displacement, added as part of effective address calculation after any memory indirection; use is optional with a size of 16 or 32 bits.

PC = Program Counter

(data) = Immediate value of 8, 16, or 32 bits

() = Effective Address

[] = Use as indirect address to long word address.

memories increased average access time. By placing a high speed cache between the processor and the rest of the memory system, the effective memory access time becomes:

$$t_{acc} = h \cdot t_{cache} + (1 - h) \cdot t_{ext}$$

where t_{acc} is the effective system access time, t_{cache} is the cache access time, t_{ext} is the access time of the rest of the system, and h is the hit ratio or the percentage of time that the data is found in the cache. Thus, for a given system design, two MC68030 on-chip caches provide an even more substantial CPU performance increase over that obtainable with the MC68020 with its instruction cache. Alternately, slower and less expensive memories can be used for the same processor performance.

The throughput increase in the MC68030 is gained in three ways. First, the MC68030 caches are accessed in less time than is required for external accesses, providing improvement in the access time for items residing in the cache. Secondly, the burst filling of the caches allows instruction and data words to be found in the on-chip caches the first time they are accessed by the micro-machine, with the time required to bring those items into the cache minimized. This has the capability of lowering the average access time for items found in the caches even further.

Thirdly, and perhaps most importantly, the autonomous nature of the caches allows instruction stream fetches, data fetches, and a third external access to all

Table 2. Instruction Set

Mnemonic	Description
ABCD ADD ADDA ADDI ADDQ ADDX AND ANDI ASL, ASR	Add Decimal with Extend Add Add Address Add Immediate Add Quick Add with Extend Logical AND Logical AND Immediate Arithmetic Shift Left and Right
Bcc BCHG BCLR BFCHG BFCLR BFEXTS BFEXTU BFFFO BFINS BFSET BFTST BKPT BRA BSET BSR BTST	Branch Conditionally Test Bit and Change Test Bit and Clear Test Bit Field and Change Test Bit Field and Clear Signed Bit Field Extract Unsigned Bit Field Extract Bit Field Find First One Bit Field Insert Test Bit Field and Set Test Bit Field Breakpoint Branch Test Bit and Set Branch to Subroutine Test Bit
CALLM CAS CAS2 CHK CHK2 CLR CMP CMPA CMPI CMPM CMP2	Call Module Compare and Swap Operands Compare and Swap Dual Operands Check Register Against Bound Check Register Against Upper and Lower Bounds Clear Compare Compare Address Compare Immediate Compare Memory to Memory Compare Register Against Upper and Lower Bounds
DBcc DIVS, DIVSL DIVU, DIVUL	Test Condition, Decrement and Branch Signed Divide Unsigned Divide
EOR EORI EXG EXT, EXTB	Logical Exclusive OR Logical Exclusive OR Immediate Exchange Registers Sign Extend
ILLEGAL	Take Illegal Instruction Trap
JMP JSR	Jump Jump to Subroutine

Mnemonic	Description
LEA LINK LSL, LSR	Load Effective Address Link and Allocate Logical Shift Left and Right
MOVE MOVEA MOVE CCR MOVE SR MOVE USP MOVEC MOVEM MOVEP MOVEQ MOVES MULS MULU	Move Move Address Move Condition Code Register Move Status Register Move User Stack Pointer Move Control Register Move Multiple Registers Move Peripheral Move Quick Move Alternate Address Sapce Signed Multiply Unsigned Multiply
NBCD NEG NEGX NOP NOT	Negate Decimal with Extend Negate Negate with Extend No Operation Logical Complement
OR ORI	Logical Inclusive OR Logical Inclusive OR Immediate
PACK PEA	Pack BCD Push Effective Address
RESET ROL, ROR ROXL, ROXR RTD RTE RTM RTR RTS	Reset External Devices Rotate Left and Right Rotate with Extend Left and Right Return and Deallocate Return from Exception Return from Module Return and Restore Codes Return from Subroutine
SBCD Scc STOP SUB SUBA SUBI SUBQ SUBX SWAP	Subtract Decimal with Extend Set Conditionally Stop Subtract Subtract Address Subtract Immediate Subtract Quick Subtract with Extend Swap Register Words
TAS TRAP TRAPcc TRAPV TST	Test Operand and Set Trap Trap Conditionally Trap on Overflow Test Operand
UNLK UNPK	Unlink Unpack BCD

MMU INSTRUCTIONS

PMOVE	Move to or from MMU Registers
PLOAD	Load Page Descriptor into ATC

PTEST	Test Translation
PFLUSH	Flush Selected ATC Entries
PFLUSHA	Flush Entire ATC

COPROCESSOR INSTRUCTIONS

cpBCC	Branch Conditionally
cpDBcc	Test Coprocessor Condition, Decrement and Branch
cpGEN	Coprocessor General Instruction

cpRESTORE	Restore Internal State of Coprocessor
cpSAVE	Save Internal State of Coprocessor
cpScc	Set Conditionally
cpTRAPcc	Trap Conditionally

occur simultaneously with instruction execution. For example, if the MC68030 requires both an instruction stream access and an external peripheral access, and the instruction is resident in the on-chip cache, the peripheral access will proceed unimpeded rather than being queued behind the instruction fetch. Additionally, if a data operand was also required, and it was resident in the data cache, it could also be accessed without hindering either the instruction access from its cache or the peripheral access external to the chip. The parallelism designed into the MC68030 also allows multiple instructions to execute concurrently so that several internal instructions (those that do not require any external accesses) could execute while the processor is performing an external access for a previous instruction.

INSTRUCTION CACHE

The instruction cache resident on the MC68030 is a 256-byte direct mapped cache organized as 16 blocks consisting of four long words per block. Each long word is independently accessible yielding 64 possible entries, with A1 selecting the correct word during an access. Thus each block has a tag field made up of the upper 24 address bits, the FC2 (supervisor/user) value, four valid bits (one for each long word entry) and the four long word entries (see Figure 5). The instruction cache is automatically filled by the MC68030 whenever a cache miss occurs and using the burst transfer capability, up to four long words can be filled in one burst. Neither the instruction or data caches can be manipulated directly by the programmer except by the use of the CACR register which provides cache clearing and cache entry clearing facilities. The caches can also be enabled/disabled through the use of this register. Finally, the system hardware can disable the on-chip caches at any time by the assertion of the $\overline{\text{CDIS}}$ signal.

DATA CACHE

The organization of the data cache is similar to that of the instruction cache as shown in Figure 6. However, the tag is composed of the upper 24 address bits, the four valid bits, and all three function code bits, explicitly specifying the address space associated with each block. The data cache employs a write-through policy with no write allocation of data writes. In other words, if a cache hit occurs on a write cycle, both the data cache and the external device are updated with the new data. If a write cycle generates a miss in the data cache, only the external device is updated and no data cache entry is replaced or allocated for that address.

OPERAND TRANSFER MECHANISMS

The MC68030 offers three different mechanisms by which data can be transferred into and out of the chip. Asynchronous bus cycles, compatible with the asynchronous bus on the MC68020, can transfer data in a minimum of three clock cycles and the amount of data transferred on each cycle is determined by the dynamic bus sizing mechanism on a cycle-by-cycle basis with the $\overline{\text{DSACKx}}$ signals. Synchronous bus cycles are terminated with the $\overline{\text{STERM}}$ (Synchronous Termination) signal and

always transfer 32-bits of data in a minimum of two clock cycles, increasing the bus bandwidth available for other bus masters, therefore increasing possible performance. Burst mode transfers can be used to fill blocks of the instruction and data caches when the MC68030 asserts $\overline{\text{CBREQ}}$ (Cache Burst Request). After completing the first cycle with $\overline{\text{STERM}}$, subsequent cycles may accept data on every clock cycle where $\overline{\text{STERM}}$ is asserted until the burst is completed. Use of this mode can further increase the available bus bandwidth in systems that use DRAMs with page, nibble, or static column mode operation.

ASYNCHRONOUS TRANSFERS

Though the MC68030 has a full 32-bit data bus, it offers the ability to automatically and dynamically downsize its bus to 8 or 16 bits if peripheral devices are unable to accommodate the entire 32 bits. This feature allows the programmer the ability to write code that is not bus-width specific. For example, long word (32 bit) accesses to peripherals may be used in the code, yet the MC68030 will transfer only the amount of data that the peripheral can manage at one time. This feature allows the peripheral to define its port size as 8, 16, or 32 bits wide and the MC68030 will dynamically size the data transfer accordingly, using multiple bus cycles when necessary. Hence, programmers are not required to program for each device port size or know the specific port size before coding; hardware designers have flexibility to choose implementations independent of software prejudices.

The dynamic bus sizing is invoked with the use of the $\overline{\text{DSACKx}}$ pins and occurs on a cycle-by-cycle basis. For example, if the processor is executing an instruction that requires the reading of a long word operand, it will attempt to read 32 bits during the first bus cycle to a long word address boundary. If the port responds that it is 32 bits wide, the MC68030 latches all 32 bits of data and continues. If the port responds that it is 16 bits wide, the MC68030 latches the 16 valid bits of data and runs another cycle to obtain the other 16 bits of data. An 8-bit port is handled similarly but with four bus read cycles. Each port is fixed in the assignment to particular sections of the data bus. However, the MC68030 has no restrictions concerning the alignment of operands in memory; long word operands need not be aligned to long word address boundaries. When misaligned data requires multiple bus cycles, the MC68030 automatically runs the minimum number of bus cycles. Instructions must still be aligned to word boundaries.

The timing of asynchronous bus cycles is also determined by the assertion of the $\overline{\text{DSACKx}}$ signals on a cycle-by-cycle basis. If the $\overline{\text{DSACKx}}$ signals are valid 1.5 clocks after the beginning of the bus cycle (with the appropriate setup time), the cycle terminates in its minimum amount of time corresponding to three clock cycles total. The cycle can be lengthened by delaying $\overline{\text{DSACKx}}$ (effectively inserting wait states in one clock increments) until the device being accessed is able to terminate the cycle. This flexibility gives the processor the ability to communicate with devices of varying speeds while operating at the fastest rate possible for each device.

Use of the asynchronous transfer mechanism allows external errors to abort cycles upon the assertion of $\overline{\text{BERR}}$ (Bus Error), or individual bus cycles to be retried with the

simultaneous assertion of $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$, after the $\overline{\text{DSACKx}}$ signals have been asserted.

SYNCHRONOUS TRANSFERS

Synchronous bus cycles are terminated with the assertion of the $\overline{\text{STERM}}$ signal which automatically indicates that the bus transfer is for 32 bits. This input is not synchronized internally thereby allowing two clock cycle bus accesses to be performed if the signal is valid one clock after the beginning of the bus cycle with the appropriate setup time. However, the bus cycle may be lengthened by delaying $\overline{\text{STERM}}$ (inserting wait states in one clock increments) until the device being accessed is able to terminate the cycle as in the case of asynchronous transfers. Additionally, these cycles may be aborted upon the asserting of $\overline{\text{BERR}}$, or they may be retried with the simultaneous assertion of $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$, after the assertion of $\overline{\text{STERM}}$.

BURST READ CYCLES

The MC68030 provides support for burst filling of its on-chip instruction and data caches, adding to the overall system performance. The on-chip caches are organized with a block size of four long words, so that there is only one tag for the four long words in a block. Since locality of reference is present to some degree in most programs, filling of all four entries when a single entry misses can be advantageous, especially if the time spent filling the additional entries is minimal. When the caches are burst-filled, data can be latched by the processor in as little as one clock for each 32 bits.

Burst read cycles can be performed only when the MC68030 requests them (with the assertion of $\overline{\text{CBREQ}}$) and only when the bus cycles are terminated with $\overline{\text{STERM}}$ as described above. If the $\overline{\text{CBACK}}$ (Cache Burst Acknowledge) input is valid at the appropriate time in the synchronous bus cycle, the processor will keep the original $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{R/W}}$, address, function code, and size outputs asserted and will latch 32 bits from the data bus at the end of each subsequent clock cycle that has $\overline{\text{STERM}}$ asserted. This procedure continues until the burst is complete (the entire block has been transferred), $\overline{\text{BERR}}$ is asserted in lieu of $\overline{\text{STERM}}$, or the $\overline{\text{CBACK}}$ input is negated.

EXCEPTIONS

KINDS OF EXCEPTIONS

Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts, the bus error, and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset pins are used for access control and processor restart. The internally generated exceptions come from instructions, address errors, tracing, or breakpoints. The TRAP , TRAPcc , TRAPVcc , cpTRAPcc , CKH , CKH2 , and DIV instructions can all generate exceptions as part of their instruction execution.

Tracing behaves like a very high priority, internally generated interrupt whenever it is processed. The other internally generated exceptions are caused by illegal instructions, instruction fetches from odd addresses, and privilege violations. Finally, the MMU can generate exceptions when it detects an invalid translation in the ATC (Address Translation Cache) and an access to the corresponding address is attempted, or when it is unable to locate a valid translation for an address in the translation tables.

EXCEPTION PROCESSING SEQUENCE

Exception processing occurs in four steps. During the first step, an internal copy is made of the status register. After the copy is made, the special processor state bits in the status register are changed. The S bit is set, putting the processor into supervisor state. Also, the T1 and T0 bits are negated, allowing the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor read that is classified as an interrupt acknowledge cycle. For coprocessor detected exceptions, the vector number is included in the coprocessor exception primitive response. For all other exceptions, internal logic provides the vector number. This vector number is then used to generate the address of the exception vector.

The third step is to save the current processor status. The exception stack frame is created and filled on the supervisor stack. In order to minimize the amount of machine state that is saved, various stack frame sizes are used to contain the processor state, depending on the type of exception and where it occurred during instruction execution. If the exception is an interrupt and the M bit is on, the M bit is forced off, and the short four word exception stack frame is saved on the master stack which indicates that the exception is saved on the interrupt stack. If the exception is a reset, the M bit is simply forced off and the reset vector is accessed.

The MC68030 provides the same extensions to the exception stacking process as the MC68020. If the M bit in the status register is set, the master stack pointer (MSP) is used for all task related exceptions. When a non-task related exception occurs (i.e., an interrupt), the M bit is cleared and the interrupt stack pointer (ISP) is used. This feature allows all the task's stack area to be carried within a single processor control block and new tasks may be initiated by simply reloading the master stack pointer and setting the M bit.

The fourth and last step of exception processing is the same for all exceptions. The exception vector offset is determined by multiplying the vector number by four. This offset is then added to the contents of the vector base register (VBR) to determine the memory address of the exception vector. The new program counter is fetched from the exception vector. The instruction at the address given in the exception vector is fetched and normal instruction decoding and execution is started.

MC68030 ON-CHIP PAGED MEMORY MANAGEMENT

The full addressing range of the MC68030 is four gigabytes (4,294,967,296 bytes). However, most MC68030 systems implement a smaller physical memory. Nonetheless, by using virtual memory techniques, the system can be made to appear to have a full four gigabytes of physical memory available to each user program. In a similar fashion, a virtual system provides user programs access to other devices that are not physically present in the system such as tape drives, disk drives, printers, or terminals. The paged Memory Management Unit (MMU) on the MC68030 provides the capability to easily support a virtual system and virtual memory. In addition, it provides protection of supervisor areas from accesses by user programs and also provides write protection on a page basis. All this capability is provided along with maximum performance as address translations occur in parallel with other processor activities. For applications not requiring the paged Memory Management Unit a register bit is used to enable/disable this feature.

DEMAND PAGED IMPLEMENTATION

A typical system with a large addressing range such as one with the MC68030 provides a limited amount of high-speed physical memory that can be accessed directly by the processor while maintaining an image of a much larger "virtual" memory on secondary storage devices such as large capacity disk drives. When the processor attempts to access a location in the virtual memory map that is not resident in physical memory, the access to that location is temporarily suspended while the necessary data is fetched from secondary storage and placed in physical memory; the suspended access is then either restarted or continued.

A paged system is one in which the physical memory is subdivided into equal sized blocks called page frames and the logical (untranslated) address space of a task is divided into pages which have the same size as the page frames. The operating system controls the allocation of pages to page frames so that when data is needed from the secondary storage device, it is brought in on a page basis. The memory management scheme employed by the MC68030 is called a "demand" implementation because a process does not need to specify in advance what areas of its logical address space it requires. An access to a logical address is interpreted by the system as a request for the corresponding page.

The MMU on the MC68030 employs the same address translation mechanism introduced by the MC68851 Paged Memory Management Unit with possible page sizes ranging from 256 bytes to 32K bytes.

TRANSLATION MECHANISM

Logical-to-physical address translation is the most frequently executed operation of the MC68030 MMU, so this task has been optimized and can function autonomously. The MMU initiates address translation by searching for a descriptor with the address translation information (a page descriptor) in the on-chip address translation cache

(ATC). The ATC is a very fast fully-associative cache memory that stores recently used page descriptors. If the descriptor does not reside in the ATC then the MMU requests external bus cycles of the bus controller to search the translation tables in physical memory. After being located, the page descriptor is loaded into the ATC and the address is correctly translated for the access, provided no exception conditions are encountered.

The status of the page in question is easily maintained in the translation tables. When a page must be brought in from a secondary storage device, the table entry can signal that this descriptor is invalid so that the table search results in an invalid descriptor being loaded into the ATC. In this way, the access to the page is aborted and the processor initiates bus error exception processing for this address. The operating system can then control the allocation of a new page in physical memory and can load the page all within the bus error handling routine.

ADDRESS TRANSLATION CACHE

An integral part of the translation function described above is the cache memory that stores recently used logical-to-physical address translation information, or page descriptors. This cache consists of 22 entries and is fully-associative. The ATC compares the logical address and function code of the incoming access against its entries. If one of the entries matches, there is a hit and the ATC sends the physical address to the bus controller, which then starts the external bus cycle (provided there was no hit in the instruction or data caches for the access).

The ATC is composed of three major components: the content-addressable memory (CAM) containing the logical address and function code information to be compared against incoming logical addresses, the physical address store that contains the physical address associated with a particular CAM entry, and the control section containing the entry replacement circuitry that implements the replacement algorithm (a variation of the least-recently-used algorithm).

TRANSLATION TABLES

The translation tables supported by the MC68030 have a tree structure, minimizing the amount of memory necessary to set up the tables for most programs, since only a portion of the complete tree needs to exist at any one time. The root of a translation table tree is pointed to by one or two root pointer registers that are part of the MC68030 programmer's model; the CPU and supervisor. Table entries at the higher levels of the tree (pointer tables) contain pointers to other tables. Entries at the leaf level (page tables) contain page descriptors. The mechanism for performing table searches uses portions of the logical address as indices for each level of the lookup. All addresses contained in the translation table entries are physical addresses.

Figure 7 illustrates the structure of the MC68030 translation tables. Several determinants of the detailed table structure are software selectable. The first level of lookup in the table normally uses the function codes as an index but this may be suppressed if desired. In addition, up to 15 of the logical address lines can be ignored for the purposes of the table searching. The number of levels in

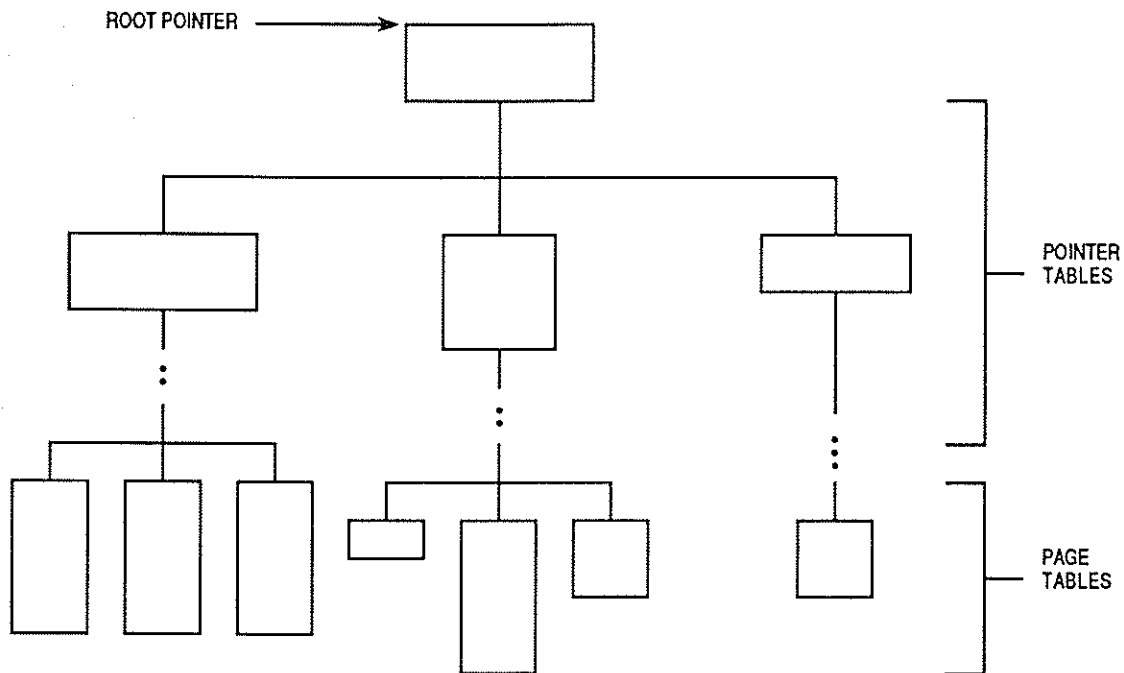


Figure 7. MMU Translation Table Tree Structure

the table indexed by the logical address can be set from one to four, and up to 15 logical address bits can be used as an index at each level. A major advantage to using this tree structure for the translation tables is the ability to deallocate large portions of the logical address space with a single entry at the higher levels of the tree. Additionally, portions of the tree itself may reside on a secondary storage device or may not exist at all until they are required by the system.

The entries in the translation tables contain status information pertaining to the pointers for the next level of lookup or the page themselves. These bits can be used to designate certain pages or blocks of pages as supervisor-only, write-protected, or non-cacheable. If a page is marked as non-cacheable, accesses within the page will not be cached by the MC68030 instruction or data caches and the CIOU (cache inhibit out) signal is asserted for those accesses. In addition, the MMU automatically maintains history information for the pointers and pages in the descriptors via the Used (U) and Modified (M) bits.

MMU INSTRUCTIONS

The MMU instructions supported by the MC68030 are the PMOVE, PTEST, PLOAD, PFLUSH, and PFLUSHA instructions and they are completely compatible with the corresponding instructions introduced by the MC68851 PMMU. Whereas the MC68851 required the coprocessor interface to execute its instructions, the MC68030 MMU instructions execute just like all other CPU instructions. All of the MMU instructions are privileged (can be executed by the supervisor only) and are summarized below:

PMOVE Used to move data to or from MMU registers.

PTEST Takes an address and function code and searches the ATC or the translation tables for the corresponding entry. The results of the search are available in the MMU status register (PSR) and is often useful in determining the cause of a fault.

PLOAD Takes an address and function code and searches the translation tables for the corresponding page descriptor. It then loads the ATC with the appropriate information.

PFLUSH Flushes the ATC by function code or function code and logical address.

PFLUSHA Flushes all of the ATC entries.

TRANSPARENT TRANSLATION

Two transparent translation registers have been provided on the MC68030 MMU to allow portions of the logical address space to be transparently mapped and accessed without corresponding entries resident in the ATC. Each register can be used to define a range of logical addresses from 16M bytes to 4G bytes with a base address and a mask. All addresses within these ranges will not be mapped and protection is provided only on a basis of read/write and function code.

COPROCESSOR INTERFACE

The coprocessor interface is a mechanism for extending the instruction set of the M68000 Family. The interface provided on the MC68030 is the same as that on the

MC68020. Examples of these extensions are the addition of specialized data operands for the existing data types or, for the case of floating point, the inclusion of new data types and operations for them as implemented by the MC68881 and MC68882 floating-point coprocessors.

Coprocessors are divided into two types by their bus utilization characteristics. A coprocessor is a DMA coprocessor if it can control the bus independent of the main processor. A coprocessor is a non-DMA coprocessor if it does not have the capability of controlling the bus. Both coprocessor types utilize the same protocol and main processor resources. Implementation of a coprocessor as a DMA or non-DMA is based primarily on bus bandwidth requirements of the coprocessor, performance, and cost issues.

The communication protocol between the main processor and the coprocessor necessary to execute a coprocessor instruction is based on a group of coprocessor interface registers (CIRs) which have been defined for the M68000 Family (see Table 3) and are implemented on the coprocessor. The MC68030 hardware uses standard read and write cycle to access the registers. Thus the coprocessor interface doesn't require any special bus hardware; the bus interface implemented by a coprocessor for its interface register set must only satisfy the MC68030 address, data, and control signal timing to guarantee proper communication with the CPU. The MC68030 implements the communication protocol with all coprocessors in hardware (and microcode) and handles all operations automatically so the programmer is only concerned with the instructions and data types provided by the coprocessor as extensions to the MC68030 instruction set and data types.

Since the CIRs are accessed via normal read and write cycles, coprocessors can be used as peripheral devices

by other M68000 Family members that do not support the coprocessor interface. The communication protocol can be easily emulated by addressing the CIRs appropriately and passing the necessary commands and operands required by the coprocessor. In addition to the CIRs, the coprocessor contains those registers added to the MC68030 programmer's model for specific coprocessor operations. For example, the Motorola floating-point coprocessors contain the CIRs as well as eight 80-bit floating-point data registers and three 32-bit control/status registers.

Up to eight coprocessors are supported in a single MC68030 system with a system-unique coprocessor identifier encoded in the coprocessor instruction. When accessing a coprocessor, the MC68030 executes standard bus cycles in CPU address space, as encoded by the function codes, and places the coprocessor identifier on the address bus to be used by chip-select logic to select the particular coprocessor. Since standard bus cycles are used, the coprocessor may be located according to system design requirements, whether it be located on the microprocessor local bus, on another board on the system bus, or any other place where the chip-select and coprocessor protocol using standard bus cycles can be supported.

COPROCESSOR PROTOCOL

Interprocessor transfers are all initiated by the main processor during coprocessor instruction execution. During the processing of a coprocessor instruction, the main processor transfers instruction information and data to the associated coprocessor, and receives data, requests, and status information from the coprocessor. These transfers are all based on standard read and write bus cycles.

The typical coprocessor protocol which the main processor follows is:

- a) The main processor initiates the communication by writing command information to a location in the coprocessor interface.
- b) The main processor reads the coprocessor response to that information.
 - 1) The response may indicate that the coprocessor is busy, and main processor should again query the coprocessor. This allows the main processor and coprocessor to synchronize their concurrent operations.
 - 2) The response may indicate some exception condition; the main processor acknowledges the exception and begins exception processing.
 - 3) The response may indicate that the coprocessor needs the main processor to perform some service such as transferring data to or from the coprocessor. The coprocessor may also request that the main processor query the coprocessor again after the service is complete.
 - 4) The response may indicate that the main processor is not needed for further processing of the instruction. The communication is terminated, and the main processor is free to begin execution of the next instruction. At this

Table 3. Coprocessor Interface Registers

Register	Function	R/W
Response	Requests Action from CPU	R
Control	CPU Directed Control	\overline{W}
Save	Initiate Save of Internal State	R
Restore	Initiate Restore of Internal State	R/ \overline{W}
Operation Word	Current Coprocessor Instruction	\overline{W}
Command Word	Coprocessor Specific Command	\overline{W}
Condition Word	Condition to be Evaluated	\overline{W}
Operand	32-Bit Operand	R/ \overline{W}
Register Select	Specifies CPU Register or Mask	R
Instruction Address	Pointer to Coprocessor Instruction	R/ \overline{W}
Operand Address	Pointer to Coprocessor Operand	R/ \overline{W}

point in the coprocessor protocol, as the main processor continues to execute the instruction stream, the main processor may operate concurrently with the coprocessor.

When the main processor encounters the next coprocessor instruction, the main processor queries the coprocessor until the coprocessor is ready; meanwhile, the main processor can go on to service interrupts and do a context switch to execute other tasks, for example.

Each coprocessor instruction type has specific requirements based on this simplified protocol. The coprocessor interface may use as many extension words as required to implement a coprocessor instruction.

PRIMITIVE/RESPONSE

The response register is the means by which the coprocessor communicates service requests to the main processor. The content of the coprocessor response register is a primitive instruction to the main processor which is read during coprocessor communication by the main processor. The main processor "executes" this primitive, thereby providing the services required by the coprocessor. Table 4 summarizes the coprocessor primitives that the MC68030 accepts.

SIGNAL DESCRIPTION

Figure 8 and Table 5 describe the signals on the MC68030 and provide an indication of their function.

Table 4. Coprocessor Primitives

Processor Synchronization Busy with Current Instruction Proceed with Next Instruction, If No Trace Service Interrupts and Re-query, If Trace Enabled Proceed with Execution, Condition True/False
Instruction Manipulation Transfer Operation Word Transfer Words from Instruction Stream
Exception Handling Take Privilege Violation if S Bit Not Set Take Pre-Instruction Exception Take Mid-Instruction Exception Take Post-Instruction Exception
General Operand Transfer Evaluate and Pass (ea) Evaluate (ea) and Transfer Data Write to Previously Evaluated (ea) Take Address and Transfer Data Transfer to/from Top of Stack
Register Transfer Transfer CPU Register Transfer CPU Control Register Transfer Multiple CPU Registers Transfer Multiple Coprocessor Registers Transfer CPU SR and/or ScanPC

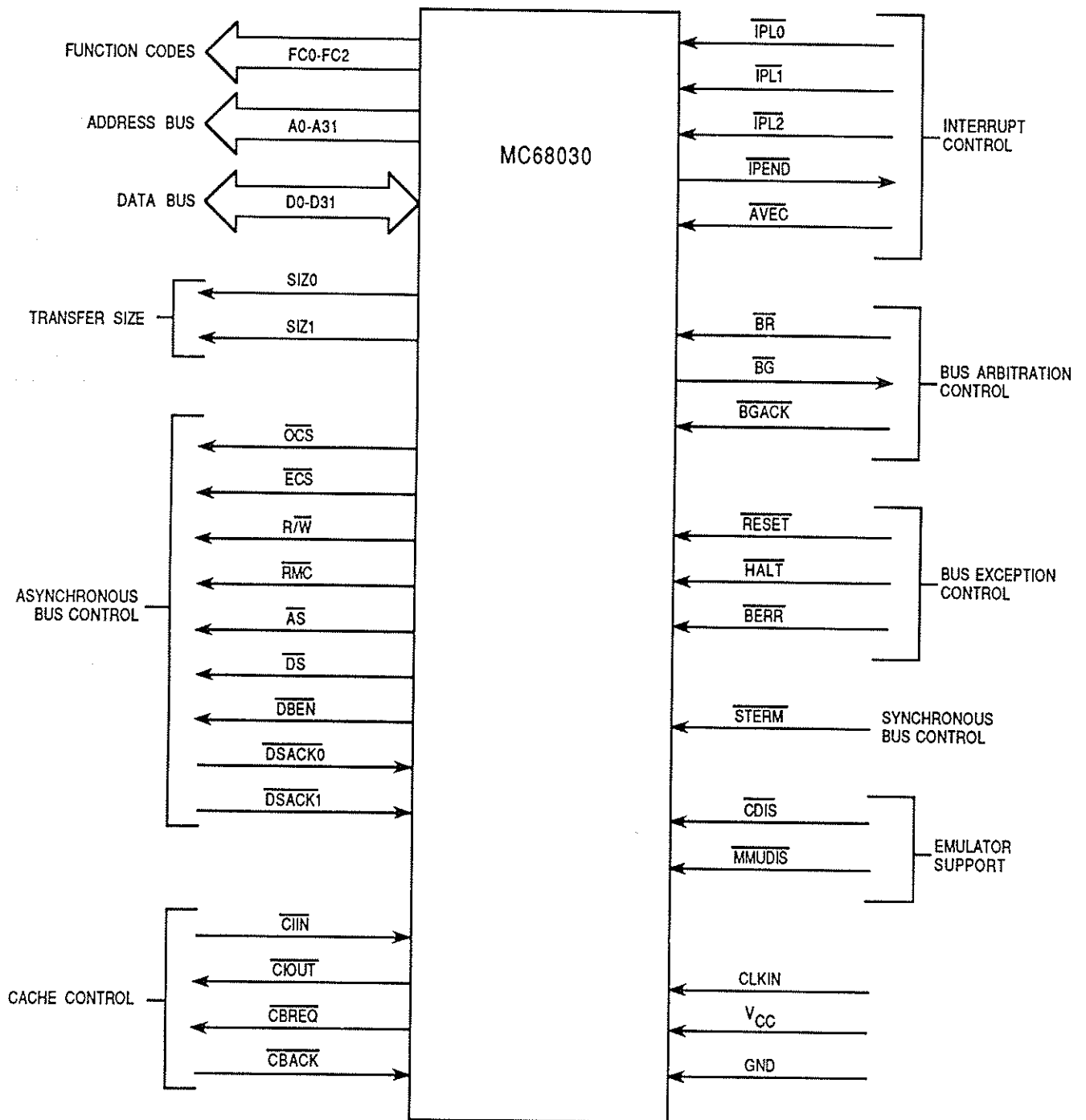


Figure 8. MC68030 Functional Signal Groups

Table 5. Signal Index

Signal Name	Mnemonic	Function
Function Codes	FC0-FC2	3-bit function code used to identify the address space of each bus cycle.
Address Bus	A0-A31	32-bit address bus used to address any of 4,294,967,296 bytes.
Data Bus	D0-D31	32-bit data bus used to transfer 8, 16, 24, or 32 bits of data per bus cycle.
Size	SIZ0/SIZ1	Indicates the number of bytes remaining to be transferred for this cycle. These signals, together with A0 and A1, define the active sections of the data bus.
Operand Cycle Start	$\overline{\text{OCS}}$	Identical operation to that of $\overline{\text{ECS}}$ except that $\overline{\text{OCS}}$ is asserted only during the first bus cycle of an operand transfer.
External Cycle Start	$\overline{\text{ECS}}$	Provides an indication that a bus cycle is beginning.
Read/Write	$\overline{\text{R/W}}$	Defines the bus transfer as an MPU read or write.
Read-Modify-Write Cycle	$\overline{\text{RMC}}$	Provides an indicator that the current bus cycle is part of an indivisible read-modify-write operation.
Address Strobe	$\overline{\text{AS}}$	Indicates that a valid address is on the bus.
Data Strobe	$\overline{\text{DS}}$	Indicates that valid data is to be placed on the data bus by an external device or has been placed on the data bus by the MC68020.
Data Buffer Enable	$\overline{\text{DBEN}}$	Provides an enable signal for external data buffers.
Data Transfer and Size Acknowledge	$\overline{\text{DSACK0/DSACK1}}$	Bus response signals that indicate the requested data transfer operation is completed. In addition, these two lines indicate the size of the external bus port on a cycle-by-cycle basis and are used for asynchronous transfers.
Cache Inhibit In	$\overline{\text{CIIN}}$	Prevents data from being loaded into the MC68030 instruction and data caches.
Cache Inhibit Out	$\overline{\text{CIOUT}}$	Reflects the CI bit in ATC entries; indicates that external caches should ignore these accesses.
Cache Burst Request	$\overline{\text{CBREQ}}$	Indicates a burst request for the instruction or data cache.
Cache Burst Acknowledge	$\overline{\text{CBACK}}$	Indicates that accessed device can operate in burst mode.
Interrupt Priority Level	$\overline{\text{IPL0-IPL2}}$	Provides an encoded interrupt level to the processor.
Interrupt Pending	$\overline{\text{IPEND}}$	Indicates that an interrupt is pending.
Autovector	$\overline{\text{AVEC}}$	Requests an autovector during an interrupt acknowledge cycle.
Bus Request	$\overline{\text{BR}}$	Indicates that an external device requires bus mastership.
Bus Grant	$\overline{\text{BG}}$	Indicates that an external device may assume bus mastership.
Bus Grant Acknowledge	$\overline{\text{BGACK}}$	Indicates that an external device has assumed bus mastership.
Reset	$\overline{\text{RESET}}$	System reset.
Halt	$\overline{\text{HALT}}$	Indicates that the processor should suspend bus activity.
Bus Error	$\overline{\text{BERR}}$	Indicates an invalid or illegal bus operation is being attempted.
Synchronous Termination	$\overline{\text{STERM}}$	Bus response signal that indicates a port size of 32 bits and that data may be latched on the next falling clock edge.
Cache Disable	$\overline{\text{CDIS}}$	Dynamically disables the on-chip cache to assist emulator support.
MMU Disable	$\overline{\text{MMUDIS}}$	Dynamically disables the translation mechanism of the MMU.
Clock	CLK	Clock input to the processor.
Power Supply	V _{CC}	+5 volt \pm 5% power supply.
Ground	GND	Ground connection.

ELECTRICAL CHARACTERISTICS

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	V
Input Voltage	V_{in}	-0.3 to +7.0	V
Operating Temperature Range	T_A	0 to 70	°C
Storage Temperature Range	T_{stg}	-55 to 150	°C

This device contains protective circuitry against damage due to high static voltages or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{CC}).

THERMAL CHARACTERISTICS — PGA PACKAGE

Characteristic	Symbol	Value	Rating
Thermal Resistance — Ceramic Junction to Ambient	θ_{JA}	30	°C/W
Junction to Case	θ_{JC}	10	

POWER CONSIDERATIONS

The average chip-junction temperature, T_J , in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

where:

T_A = Ambient Temperature, °C

θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D = P_{INT} + P_{I/O}$

$P_{INT} = I_{CC} \times V_{CC}$, Watts — Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins — User Determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273^\circ\text{C}) \quad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_D \cdot (T_A + 273^\circ\text{C}) + \theta_{JA} \cdot P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A . Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A .

The total thermal resistance of a package (θ_{JA}) can be separated into two components, θ_{JC} and θ_{CA} , representing the barrier to heat flow from the semiconductor junction to the package (case) surface (θ_{JC}) and from the case to the outside ambient (θ_{CA}). These terms are related by the equation:

$$\theta_{JA} = \theta_{JC} + \theta_{CA} \quad (4)$$

θ_{JC} is device related and cannot be influenced by the user. However, θ_{CA} is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling and thermal convection. Thus, good thermal management on the part of the user can significantly reduce θ_{CA} so that θ_{JA} approximately equals θ_{JC} . Substitution of θ_{JC} for θ_{JA} in equation (1) will result in a lower semiconductor junction temperature.

Values for thermal resistance presented in this document, unless estimated, were derived using the procedure described in Motorola Reliability Report 7843, "Thermal Resistance Measurement Method for MC68XX Microcomponent Devices," and are provided for design purposes only. Thermal measurements are complex and dependent on procedure and setup. User derived values for thermal resistance may differ.

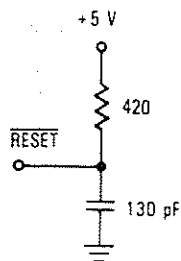


Figure 9. RESET Test Load

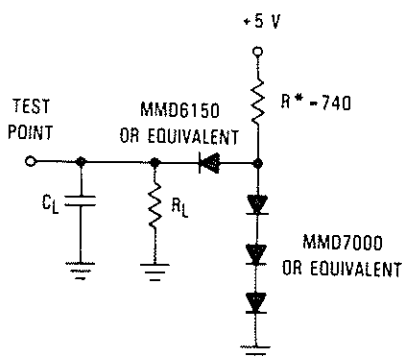


Figure 10. Test Loads

$C_L = 50 \text{ pF}$ for \overline{ECS} and \overline{OCS}

$C_L = 70 \text{ pF}$ for \overline{CIOUT}

$C_L = 130 \text{ pF}$ for All Other
(Includes all Parasitics)

$R_L = 6.0 \text{ kohm}$

$R^* = 1.22 \text{ kohm}$ for A0-A31, D0-D31,
 \overline{BG} , FC0-FC2, SIZ0-SIZ1

$R = 2 \text{ kohm}$ for \overline{ECS} , \overline{OCS} , \overline{CIOUT}

$R = 740 \text{ ohm}$ for \overline{AS} , \overline{DS} , $\overline{R/W}$, \overline{RMC} ,
 \overline{DBEN} , \overline{IPEND} , \overline{CBREQ}

DC ELECTRICAL CHARACTERISTICS ($V_{CC} = 5.0 \text{ Vdc} \pm 5\%$; $GND = 0 \text{ Vdc}$; $T_A = 0 \text{ to } 70^\circ\text{C}$; see Figures 9 and 10)

Characteristic	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	2.0	V_{CC}	V
Input Low Voltage	V_{IL}	GND -0.5	0.8	V
Input Leakage Current $GND \leq V_{in} \leq V_{CC}$ \overline{BERR} , \overline{BR} , \overline{BGACK} , CLK , $\overline{IPL0-IPL2}$, \overline{AVEC} , \overline{CDIS} , $\overline{DSACK0}$, $\overline{DSACK1}$, \overline{HALT} , \overline{RESET}	I_{in}	-2.5 -20	2.5 20	μA
Hi-Z (Off-State) Leakage Current @ 2.4 V/0.5 V A0-A31, \overline{AS} , \overline{DBEN} , \overline{DS} , D0-D31, FC0-FC2, R/W, \overline{RMC} , SIZ0-SIZ1	I_{TSI}	-20	20	μA
Output High Voltage \overline{IPEND} , $I_{OH} = 400 \mu\text{A}$ A0-A31, \overline{AS} , \overline{BG} , D0-D31, \overline{DBEN} , \overline{DS} , \overline{ECS} , R/W, \overline{IPEND} , \overline{OCS} , \overline{RMC} , SIZ0-SIZ1, FC0-FC2, \overline{CBREQ} , \overline{CIOUT}	V_{OH}	2.4	—	V
Output Low Voltage $I_{OL} = 3.2 \text{ mA}$ $I_{OL} = 5.3 \text{ mA}$ $I_{OL} = 2.0 \text{ mA}$ $I_{OL} = 10.7 \text{ mA}$ A0-A31, FC0-FC2, SIZ0-SIZ1, \overline{BG} , D0-D31 \overline{CBREQ} , \overline{AS} , \overline{DS} , R/W, \overline{RMC} , \overline{DBEN} , \overline{IPEND} , \overline{CIOUT} , \overline{ECS} , \overline{OCS} , \overline{RESET}	V_{OL}	— — — —	0.5 0.5 0.5 0.5	V
Power Dissipation ($T_A = 0^\circ\text{C}$)	P_D	—	2.6	W
Capacitance (see Note 1) $V_{in} = 0 \text{ V}$, $T_A = 25^\circ\text{C}$, $f = 1 \text{ MHz}$	C_{in}	—	20	pF

NOTE 1. Capacitance is periodically sampled rather than 100% tested.

AC ELECTRICAL SPECIFICATIONS — CLOCK INPUT (see Figure 11)

Num.	Characteristic	16.67 MHz		20 MHz		Unit
		Min	Max	Min	Max	
	Frequency of Operation	12.5	16.67	12.5	20	MHz
1	Cycle Time CLK	60	125	50	80	ns
2, 3	CLK Pulse Width	24	95	20	54	ns
4, 5	CLK Rise and Fall Times	—	5	—	5	ns

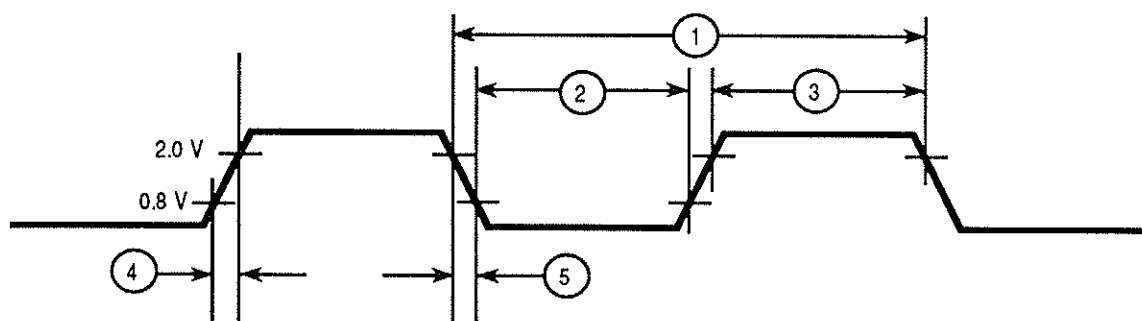


Figure 11. Clock Input Timing Diagram

AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES ($V_{CC}=5.0\text{ Vdc} \pm 5\%$; $GND=0\text{ Vdc}$; $T_A=0\text{ to }70^\circ\text{C}$;
see Figures 13 through 18)

Num.	Characteristic	16.67 MHz		20 MHz		Unit
		Min	Max	Min	Max	
6	CLK High to Function Code, Size, \overline{RMC} , \overline{CIOUT} , Address Valid	0	30	0	25	ns
6A	CLK High to \overline{ECS} , \overline{OCS} Asserted	0	20	0	15	ns
7	CLK High to Function Code, Size, \overline{RMC} , \overline{CIOUT} , Address, Data High Impedance	0	60	0	50	ns
8	CLK High to Function Code, Size, \overline{RMC} , \overline{CIOUT} , Address Invalid	0	—	0	—	ns
9 ¹⁴	CLK Low to \overline{AS} , \overline{DS} Asserted, \overline{CBREQ} Valid	3	30	3	25	ns
9A ¹	\overline{AS} to \overline{DS} Assertion Skew (Read)	-15	15	-10	10	ns
10	\overline{ECS} Width Asserted	20	—	15	—	ns
10A	\overline{OCS} Width Asserted	20	—	15	—	ns
10B ⁷	\overline{ECS} , \overline{OCS} Width Negated	15	—	10	—	ns
11 ⁶	Function Code, Size, \overline{RMC} , \overline{CIOUT} , Address Valid to \overline{AS} Asserted (and \overline{DS} Asserted, Read)	15	—	10	—	ns
12	CLK Low to \overline{AS} , \overline{DS} , \overline{CBREQ} Negated	0	30	0	25	ns
12A	CLK Low to \overline{ECS} , \overline{OCS} Negated	0	30	0	25	ns
13	\overline{AS} , \overline{DS} Negated to Function Code, Size, \overline{RMC} , \overline{CIOUT} , Address Invalid	15	—	10	—	ns
14	\overline{AS} (and \overline{DS} Read) Width Asserted (Asynchronous Cycle)	100	—	85	—	ns
14A ¹¹	\overline{DS} Width Asserted Write	40	—	38	—	ns
14B	\overline{AS} (and \overline{DS} Read) Width Asserted (Synchronous Cycle)	40	—	35	—	ns
15	\overline{AS} , \overline{DS} Width Negated	40	—	38	—	ns
15A ⁸	\overline{DS} Negated to \overline{AS} Asserted	35	—	30	—	ns
16	CLK High to \overline{AS} , \overline{DS} , R/\overline{W} , \overline{DBEN} , \overline{CBREQ} High Impedance	—	60	—	50	ns
17 ⁶	\overline{AS} , \overline{DS} Negated to R/\overline{W} Invalid	15	—	10	—	ns
18	CLK High to R/\overline{W} High	0	30	0	25	ns
20	CLK High to R/\overline{W} Low	0	30	0	25	ns
21 ⁶	R/\overline{W} High to \overline{AS} Asserted	15	—	10	—	ns
22 ⁶	R/\overline{W} Low to \overline{DS} Asserted (Write)	75	—	60	—	ns
23	CLK High to Data Out Valid	—	30	—	25	ns
25 ^{6 11}	\overline{DS} Negated to Data Out Invalid	15	—	10	—	ns
25A ^{9 11}	\overline{DS} Negated to \overline{DBEN} Negated (Write)	15	—	10	—	ns
26 ^{6 11}	Data Out Valid to \overline{DS} Asserted (Write)	15	—	10	—	ns
27	Data-In Valid to CLK Low (Setup)	5	—	5	—	ns
27A	Late $\overline{BERR}/\overline{HALT}$ Asserted to CLK Low (Setup)	15	—	10	—	ns
28 ¹²	\overline{AS} , \overline{DS} Negated to \overline{DSACKx} , \overline{BERR} , \overline{HALT} , \overline{AVEC} Negated (Asynchronous Hold)	0	60	0	50	ns
28A ¹²	CLK Low to \overline{BERR} , \overline{HALT} , \overline{AVEC} Negated (Synchronous Hold)	15	100	15	85	ns
29 ¹²	\overline{DS} Negated to Data Invalid (Asynchronous Hold)	0	—	0	—	ns
29A ¹²	\overline{DS} Negated to Data-In High Impedance	—	60	—	50	ns
30 ¹²	CLK Low to Data-In Invalid (Synchronous Hold)	15	—	15	—	ns
30A ¹²	CLK Low to Data-In High Impedance (Read followed by Write)	—	90	—	75	ns
31 ²	\overline{DSACKx} Asserted to Data-In Valid	—	50	—	43	ns
31A ³	\overline{DSACKx} Asserted to \overline{DSACKx} Valid (Skew)	—	15	—	10	ns
32	RESET Input Transition Time	—	1.5	—	1.5	Clks
33	CLK Low to \overline{BG} Asserted	0	30	0	25	ns

AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES (Continued)

Num.	Characteristic	16.67 MHz		20 MHz		Unit
		Min	Max	Min	Max	
34	CLK Low to \overline{BG} Negated	0	30	0	25	ns
35	\overline{BR} Asserted to \overline{BG} Asserted (\overline{RMC} Not Asserted)	1.5	3.5	1.5	3.5	Clks
37	\overline{BGACK} Asserted to \overline{BG} Negated	1.5	3.5	1.5	3.5	Clks
39	\overline{BG} Width Negated	90	—	75	—	ns
39A	\overline{BG} Width Asserted	90	—	75	—	ns
40	CLK High to \overline{DBEN} Asserted (Read)	0	30	0	25	ns
41	CLK Low to \overline{DBEN} Negated (Read)	0	30	0	25	ns
42	CLK Low to \overline{DBEN} Asserted (Write)	0	30	0	25	ns
43	CLK High to \overline{DBEN} Negated (Write)	0	30	0	25	ns
44 ⁶	R/W Low to \overline{DBEN} Asserted (Write)	15	—	10	—	ns
45 ⁵	\overline{DBEN} Width Asserted	Asynchronous Read		60	—	ns
		Asynchronous Write		120	—	
45A ⁹	\overline{DBEN} Width Asserted	Synchronous Read		10	—	ns
		Synchronous Write		60	—	
46	R/W Width Asserted (Asynchronous Write or Read)	150	—	125	—	ns
46A	R/W Width Asserted (Synchronous Write or Read)	90	—	75	—	ns
47A	Asynchronous Input Setup Time to CLK	5	—	5	—	ns
47B	Asynchronous Input Hold Time from CLK	15	—	15	—	ns
48 ⁴	\overline{DSACKx} Asserted to $\overline{BERR}/\overline{HALT}$ Asserted	—	30	—	20	ns
53	Data Out Hold from CLK High	0	—	0	—	ns
55	R/W Asserted to Data Bus Impedance Change	30	—	25	—	ns
56	\overline{RESET} Pulse Width (Reset Instruction)	512	—	512	—	Clks
57	\overline{BERR} Negated to \overline{HALT} Negated (Rerun)	0	—	0	—	ns
58 ¹⁰	\overline{BGACK} Negated to Bus Driven	1	—	1	—	Clks
59 ¹⁰	\overline{BG} Negated to Bus Driven	1	—	1	—	Clks
60 ¹³	Synchronous Input Valid to CLK Transition (Setup Time)	0	—	0	—	ns
61 ¹³	CLK Transition to Synchronous Input Invalid (Hold Time)	20	—	20	—	ns

NOTES:

1. This number can be reduced to 5 nanoseconds if strobes have equal loads.
2. If the asynchronous setup time (#47A) requirements are satisfied, the \overline{DSACKx} low to data setup time (#31) and \overline{DSACKx} low to \overline{BERR} low setup time (#48) can be ignored. The data must only satisfy the data-in to clock low setup time (#27) for the following clock cycle, \overline{BERR} must only satisfy the late \overline{BERR} low to clock low setup time (#27A) for the following clock cycle.
3. This parameter specifies the maximum allowable skew between $\overline{DSACK0}$ to $\overline{DSACK1}$ asserted or $\overline{DSACK1}$ to $\overline{DSACK0}$ asserted; specification #47A must be met by $\overline{DSACK0}$ or $\overline{DSACK1}$.
4. In the absence of \overline{DSACKx} , \overline{BERR} is an asynchronous input using the asynchronous input setup time (#47A).
5. \overline{DBEN} may stay asserted on consecutive write cycles.
6. Actual value depends on the clock input waveform.
7. This specification indicates the minimum high time for \overline{ECS} and \overline{OCS} in the event of an internal cache hit followed immediately by a cache miss or operand cycle.
8. This specification guarantees operation with the MC68881, which specifies a minimum time for \overline{DS} negated to \overline{AS} asserted (specification #13A in the *MC68881 User's Manual*). Without this specification, incorrect interpretation of specifications #9A and #15 would indicate that the MC68030 does not meet the MC68881 requirements.
9. This specification allows a system designer to guarantee data hold times on the output side of data buffers that have output enable signals generated with \overline{DBEN} . The timing on \overline{DBEN} precludes its use for synchronous READ cycles with no wait states.
10. These specifications allow system designers to guarantee that an alternate bus master has stopped driving the bus when the MC68030 regains control of the bus after an arbitration sequence.
11. \overline{DS} will not be asserted for synchronous write cycles with no wait states.
12. These hold times are specified with respect to strobes (asynchronous) and with respect to the clock (synchronous). The designer is free to use either hold time.
13. Synchronous inputs must meet specifications #60 and #61 with stable logic levels for *all* rising edges of the clock.
14. If the loading on these signals is reduced to 70 pF the maximum specification timing can be reduced by 7 ns.

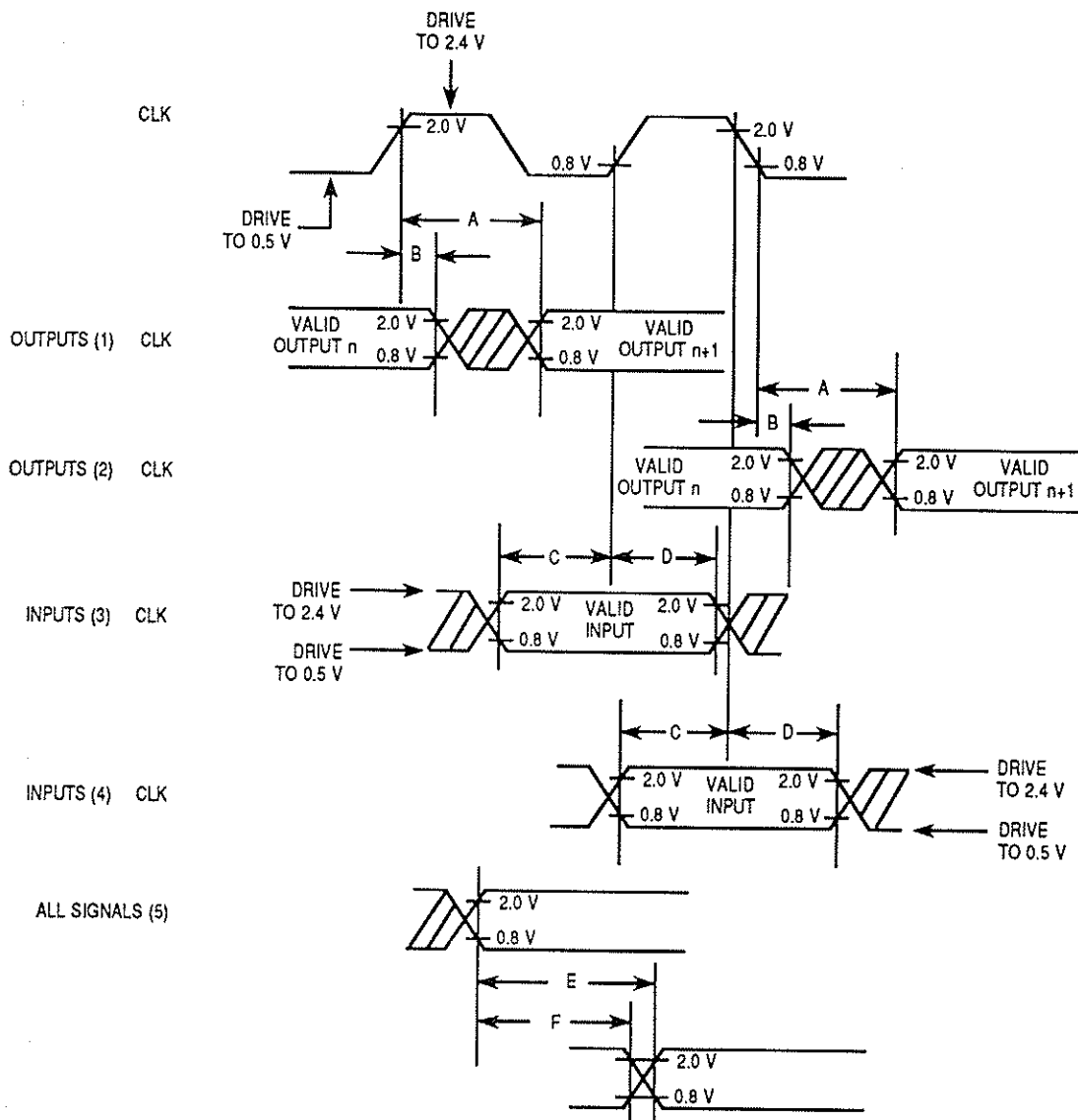
AC ELECTRICAL SPECIFICATIONS DEFINITIONS

The AC specifications presented consist of output delays, input setup and hold times, and signal skew times. All signals are specified relative to an appropriate edge of the MC68030 clock input and, possibly, relative to one or more other signals.

The measurement of the AC specifications is defined by the waveforms in Figure 12. In order to test the parameters guaranteed by Motorola, inputs must be driven to the voltage levels specified in Figure 12. Outputs of

the MC68030 are specified with minimum and/or maximum limits, as appropriate, and are measured as shown. Inputs to the MC68030 are specified with minimum and, as appropriate, maximum setup and hold times, and are measured as shown. Finally, the measurements for signal-to-signal specifications are also shown.

Note that the testing levels used to verify conformance of the MC68030 to the AC specifications does not affect the guaranteed DC operation of the device as specified in the DC electrical characteristics.



NOTES:

- 1 - This output timing is applicable to all parameters specified relative to the rising edge of the clock
- 2 - This output timing is applicable to all parameters specified relative to the falling edge of the clock
- 3 - This input timing is applicable to all parameters specified relative to the rising edge of the clock
- 4 - This input timing is applicable to all parameters specified relative to the falling edge of the clock
- 5 - This timing is applicable to all parameters specified relative to the assertion/negation of another signal

LEGEND:

- A - Maximum output delay specification
- B - Minimum output hold time
- C - Minimum input setup time specification
- D - Minimum input hold time specification
- E - Signal valid to signal valid specification (maximum or minimum)
- F - Signal valid to signal invalid specification (maximum or minimum)

Figure 12. Drive Levels and Test Points for AC Specifications

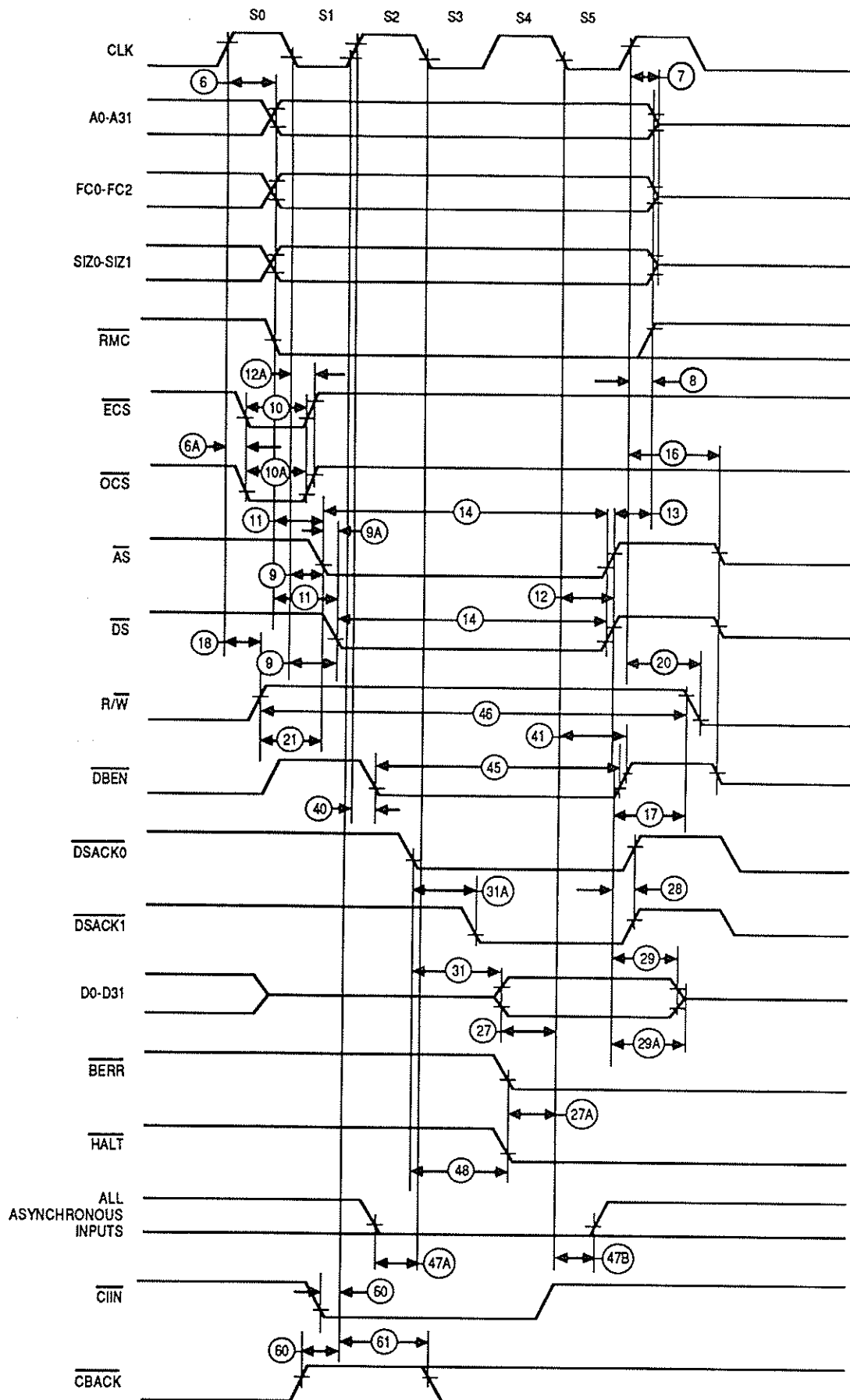


Figure 13. Asynchronous Read Cycle Timing Diagram

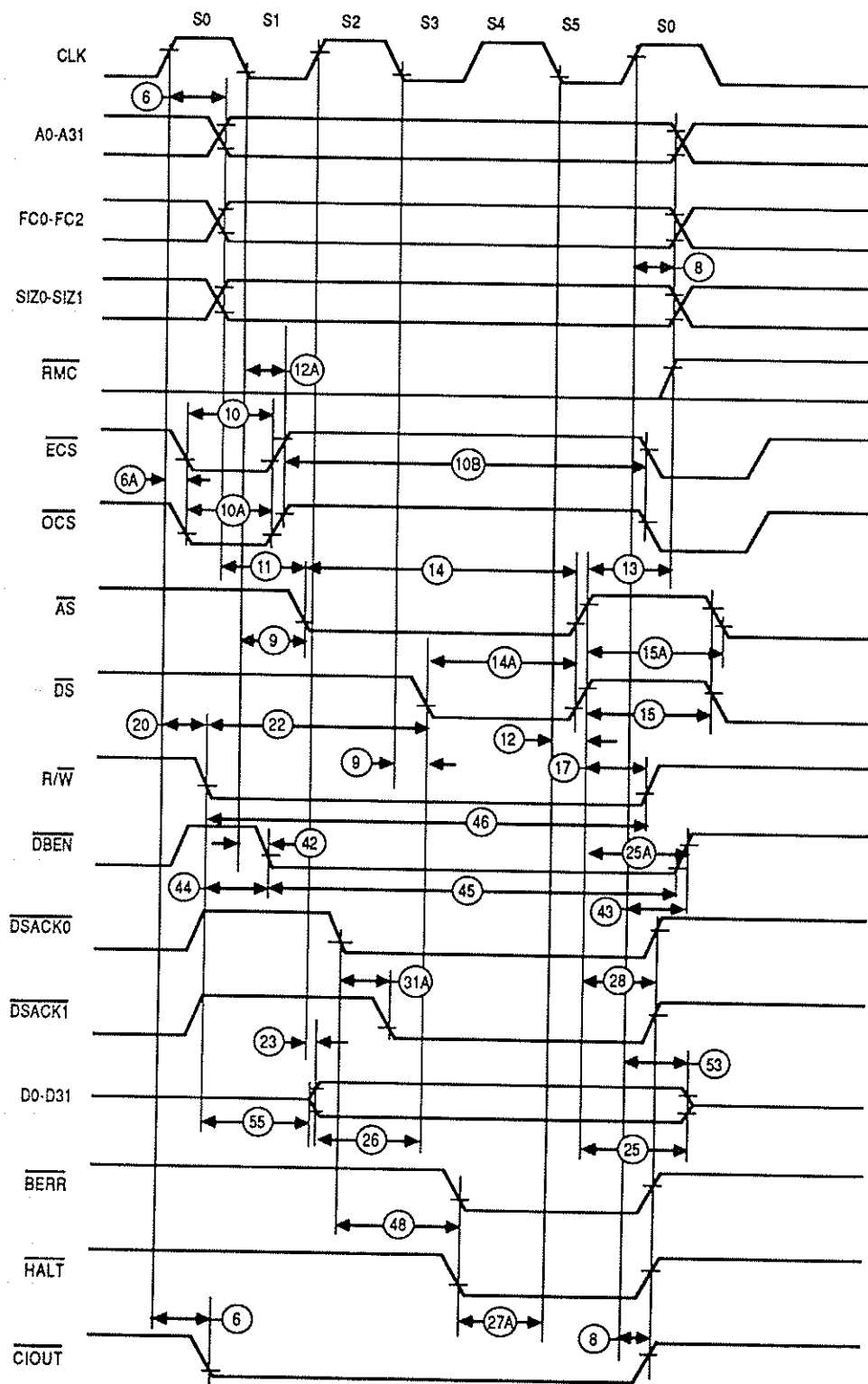


Figure 14. Asynchronous Write Cycle Timing Diagram

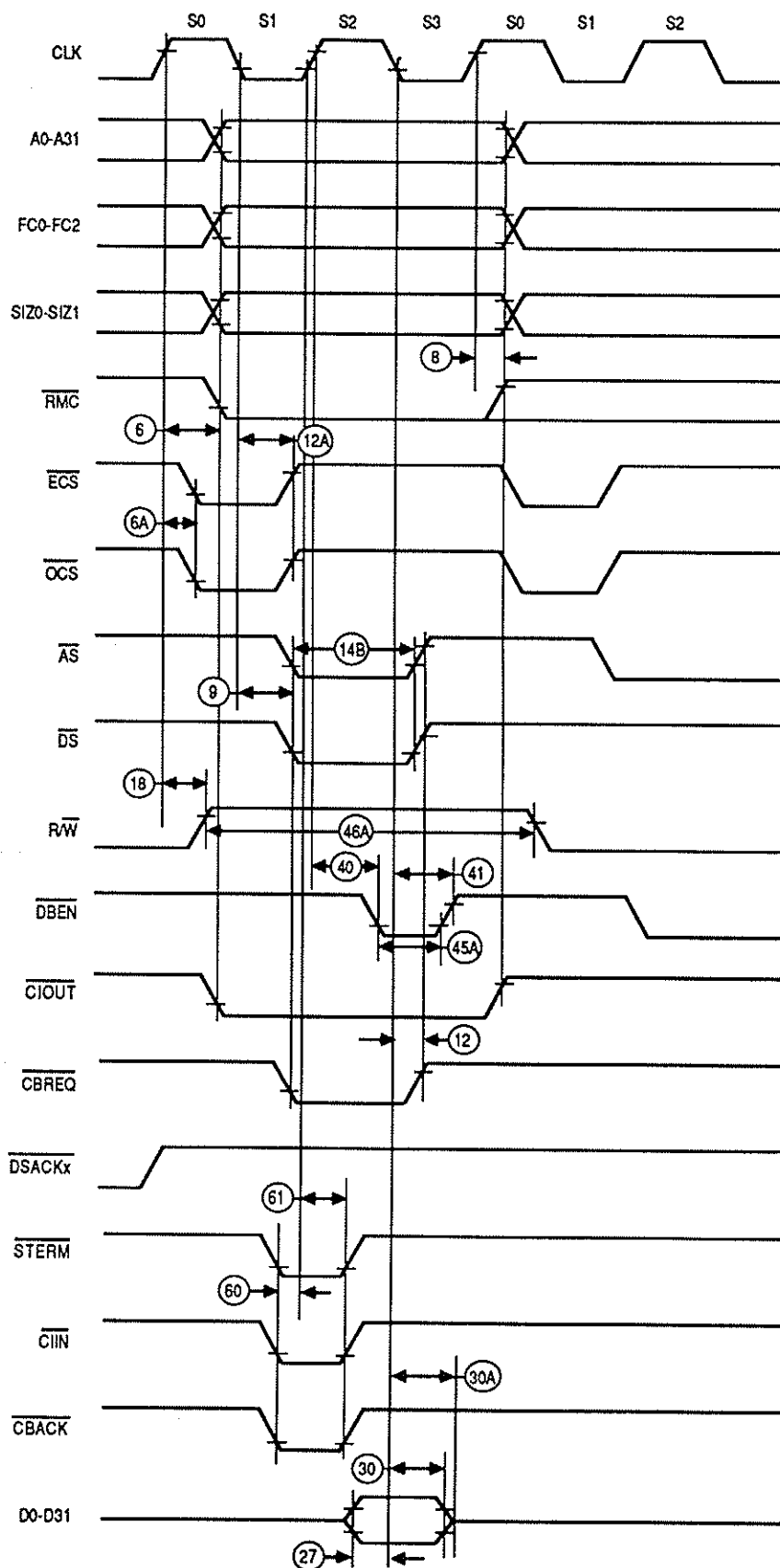


Figure 15. Synchronous Read Cycle Timing Diagram

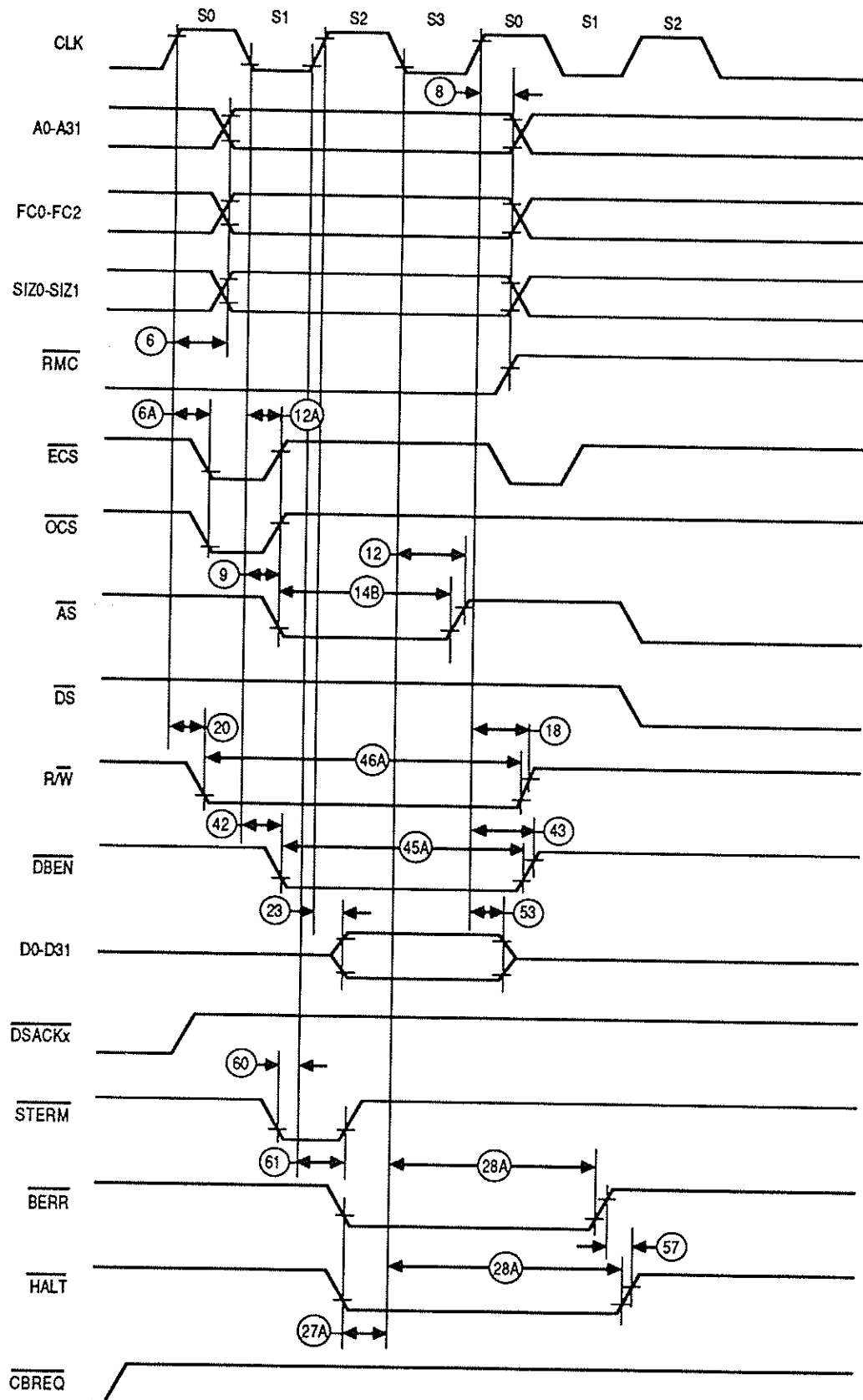


Figure 16. Synchronous Write Cycle Timing Diagram

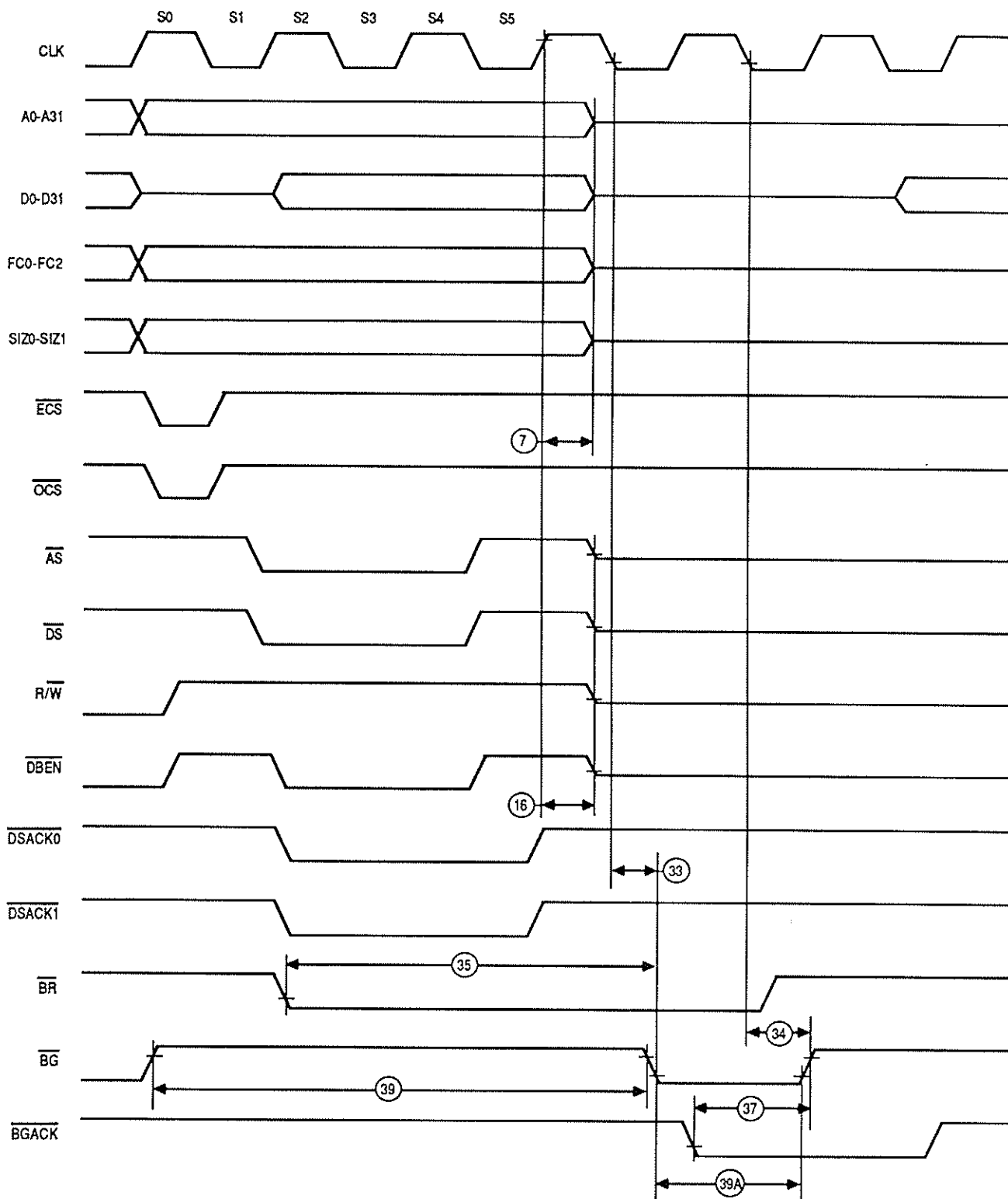


Figure 17. Bus Arbitration Timing Diagram

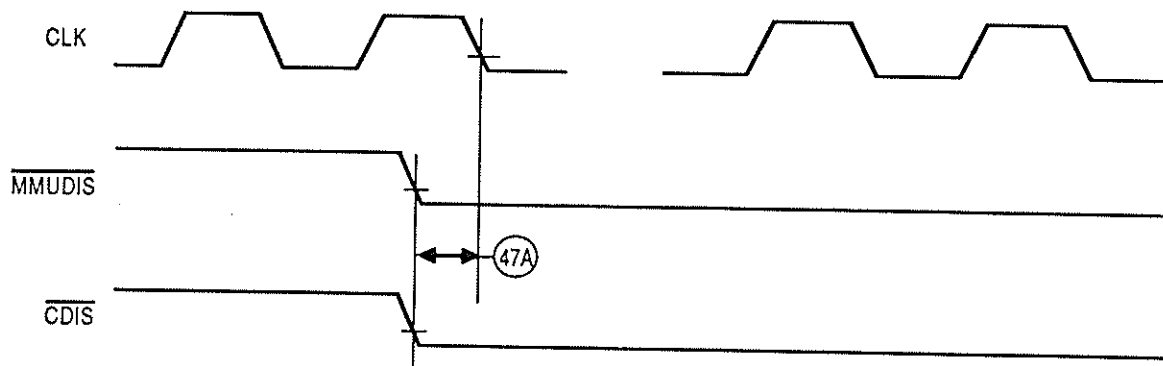



Figure 18. Other Asynchronous Signal Timings

ELECTRONICS SHOWCASE
 M MCLEAN ESQ
 CHIEF ENGINEER
 MICROCENTRE (CMS) LTD
 30 DUNDAS ST
 EDINBURGH SCOTLAND
 EH3 6TN
 (Tel: 031 556 7354)

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola and  are registered trademarks of Motorola, Inc.



MOTOROLA LTD. Semiconductor Products Group

COLVILLES ROAD, KELVIN ESTATE, EAST KILBRIDE, GLASGOW, G75 0TG, SCOTLAND